CHAPTER 1

Introducing FileMaker

FileMaker Pro is a relational data management system published by Apple subsidiary FileMaker Inc.

Since its origin, the product has expanded and evolved with improved capabilities to keep pace with standard technologies. The current incarnation is a powerful, feature-rich and easy-to-use integrated development tool, available in fifteen different languages. FileMaker databases can be shared across a network with multiple simultaneous users working on Mac and Windows computers as well as iOS devices.

FileMaker is popular with millions of developers and users, including independent consultants, employees of small businesses, and members of teams working at medium to large businesses, non-profits, and government agencies. Beginners are empowered by the ease of use and approachable interface. Professionals are won over by its rapid development capabilities for prototyping and the ability to build powerful user-friendly solutions. Business owners who don't have time to learn to program can easily find a professional consultant to develop a system tailored to meet their individual needs.

Whatever your skill set, if you are frustrated with an existing database system or are looking to develop something new, FileMaker is an excellent choice.

In this chapter, we introduce FileMaker by discussing topics such as the following:

- The history of FileMaker in a Nutshell
- Exploring database basics
- Identifying different database architectures
- Taking a bird's eye anatomical and functional overview
- The FileMaker product family

The History of FileMaker in a Nutshell

The early history of FileMaker is a zigzag between various publishers, and it employed a rather haphazard naming and numbering system before finally settling into its consistent and logical modern track.

Nashoba Nutshell

FileMaker started its life in the early 1980s as *Nutshell*, an MS-DOS computer program that was developed by Nashoba Systems in Concord, Massachusetts, and distributed by electronics marketer Leading Edge.

M. C. Munro, *Learn FileMaker Pro 16*, DOI 10.1007/978-1-4842-2863-0_1

Early FileMaker

When the Macintosh computer was introduced in early 1984, Nashoba saw an opportunity. They worked to combine the database engine with a graphic user interface and created a new forms-based database product. When Leading Edge announced that they wished to remain a DOS-only vendor, Nashoba began working with a new distributor. Forethought, Inc. FileMaker v1.0 was released in April 1985 for the Macintosh platform. In 1986, it was renamed FileMaker Plus to match the release of the Macintosh Plus.

By today's standards, the early versions of FileMaker were incredibly crude and limited in capabilities. However, for the time, the software filled an important need with an easy-to-use design and steadily became popular with the do-it-yourself crowd.

FileMaker 4

When Microsoft purchased Forethought, they tried to negotiate a purchase of FileMaker, which was outselling their own Microsoft File database application. However, Nashoba declined and decided to begin self-publishing the program, now named FileMaker 4.

Claris

In 1987, Apple formed Claris Corp. as a wholly owned subsidiary to develop and publish Macintosh software titles such as MacWrite and MacPaint. In 1988, Claris purchased Nashoba Systems to acquire FileMaker. By this time Leading Edge and Nutshell had disappeared as other DOS and Windows database products dominated the market.

In 1988, FileMaker II was released by Claris to match the naming scheme of their other products. After a few minor updates, the product was rebranded in 1990 under its modern naming format when FileMaker Pro version 1.0 was released.

Claris upgraded the product in 1992 with Windows support, making FileMaker Pro 2.0 the first cross-platform version. They began publishing a server application in 1994. It wasn't until 1995 with the release of version 3.0 that FileMaker became fully relational.

By 1997, with version 4.0, FileMaker was a widely popular product and was outselling all other Claris products. Apple absorbed those products in-house and renamed the subsidiary after their only remaining product.

FileMaker Inc.

FileMaker Inc., the newly renamed subsidiary, released FileMaker Pro 4.1v2 in June of 1999. Since that time, they have steadily improved and expanded the product. Some highlights are included in Table 1-1.

Table 1-1. *A summary of major new features in recent FileMaker Pro versions*

Year	Version	Feature Improvements
2001	5.5	Native support for Mac OS X
2004	7.0	Multiple table file architecture, multiple windows, relationship graph, calculation variables
2005	8.0	Tabs on layouts, script variables
2006	8.5	Web Viewer, layout object names
2007	9.0	SQL support, conditional formatting of layout objects
2009	10	Script triggering
2010	11.0	Charting, snapshot link, filtered portals, first version of FileMaker Go for iOS released
2012	12.0	Themes, floating and modal windows, `ExecuteSQL` function introduced, enhanced container field, improved charting
2013	13.0	WebDirect and HTML5 features; enhanced interface, hide layout object calculation, enhanced `ExecuteSQL` function, `Perform Script on Server` script step
2015	14.0	Script workspace, Button Bar layout tools, Launch Center
2016	15.0	Enhanced script workspace, concealed edit box
2017	16.0	Layout Objects window, copy/paste value lists, JSON support, enhanced security, enhanced script steps, built-in function changes, variable support in data sources, extended privilege changes

Since its start in 1985 as a modest, single-user database, the product has greatly improved. What began as a simple, easy-to-use tool for non-technical people has matured into a considerably more sophisticated platform. Today, FileMaker offers an astonishing array of technical features and capabilities that are sure to fill the most advanced development needs. By integrating this power with an intuitive graphical interface, it continues to be approachable by those new to programming.

Exploring Database Basics

Before getting into the specifics of FileMaker, let's review a few basic database concepts.

Defining a Database

A *database* is a structured collection of information efficiently stored in a generic format that makes it easily accessibility for reuse in a variety of different ways.

Most applications are focused on one specific data type with predefined properties and procedures. A calendar application allows a user to create and manage events with properties of date, time, attendees, notes, and alerts. Similarly, an email application allows management of messages with predefined properties of sender, recipients, subject, and body. These could each be thought of colloquially as an "event-base" and a "message-base" since, at their *base*, they store and provide access to *events* and *messages*.

From a wider perspective, both events and messages are types of information; they are both forms of *data*. A calendar app manages an event *database*. A mail application manages a message *database*. Unlike these, an open-ended database application like FileMaker Pro has no predefined data type; at its base is data, *any data*. It allows a developer-user to create a custom management system for *any* kind of data depending on a specific custom need.

Specific applications, such as the calendar or mail app mentioned above, provide a metaphorical filing cabinet that is locked and preconfigured to fit only certain predefined folders containing a specific type of information. A user is free to enter information into this predefined framework but have little or no control over the framework itself. By contrast, a database is the metaphorical equivalent of that same filing drawer completely empty and unlocked so that *you* can define the content it accepts, how that information is stored, related, displayed, used and shared. Every project starts fresh with the same blank slate waiting for you to define the framework and establish the capabilities it will make available to the user.

Databases can store information about *anything* a developer defines them to manage: companies, messages, notes, people, products, tasks, etc.

Database Structure

Fundamentally, a database is made up of only a few basic components. These include tables, relationships, a user interface, and scripted procedures.

A *table* can be visualized as a set of rows and columns in a spreadsheet tab that is dedicated to a single type of entity. The simple table example, shown in Figure 1-1, is modeled after `People`.

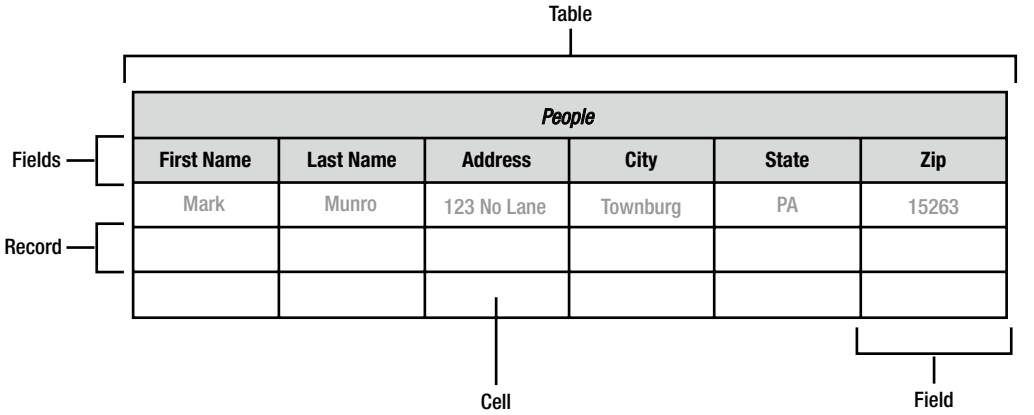

Figure 1-1. *The basic anatomy of a database table*

A *field*, represented by a column in the table, is a defined container in which one piece of information about the entity modeled will be stored. In the example shown, each component of a name and address is an individual field.

A *record*, represented by a row in the table, is one individual entity stored within a table. While the table in the example represents people in general, a row, or record, represents one person. The record is one instance of the table's defined field set containing information about a specific person.

A *cell* is the formal name of an intersection of a field and record. In FileMaker, these are commonly referred to as *fields* with the understanding that each is an instance of a field for a record (cell), which is distinct from the field definition itself (field).

Relational Databases

A database is considered *relational* when its tables can be interconnected to form relationships between the records they contain based on one or more key fields. While any field can act as a key, most often a unique identification number is assigned to each record, which is used to locate matching values in a field in another table.

A relationship allows information from one table to be accessed from the other through the connection. They allow data to be stored in separate tables but used elsewhere and help to eliminate redundant data entry.

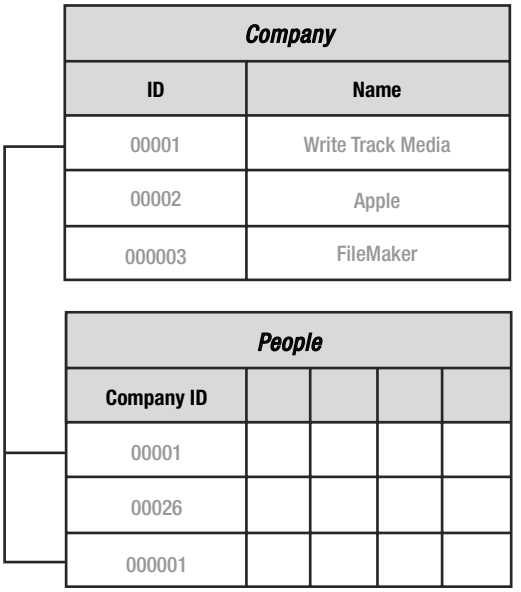

Figure 1-2. *An example relationship established between two tables*

A *relationship, like the one shown in Figure 1-2*, is a connection between two tables established by linking one or more fields from one table to one or more fields in another. In this example, the unique `ID` field in the `Company` table is linked to the `Company ID` field in the `People` table forming the relationship. We see that there are two matches, meaning two people have been linked to the same company. Using the relationship, the two-related people can be displayed on a layout for the company and used in calculations and scripted procedures from within that context.

User Interface

The *user interface* is made up of the layouts, menus, and windows that allows the user to view and interact with the data in a graphical, human-friendly format.

A *layout* is a visual display of fields and controls from the perspective of a single table's records. Tables can have many layouts for different purposes. For example, a *list* layout might show a couple fields for all the records for a table to allow faster navigational scrolling. Another layout might show every field for a single record, optimized for *data entry*.

The *current layout* is whichever layout is being displayed in the database window at a given time. This layout can be changed to another by the user manually selecting a different one or by a script. Figure 1-3 shows a database with two tables, each with a list and entry layout. The window is shown currently displaying the `People Entry` layout.

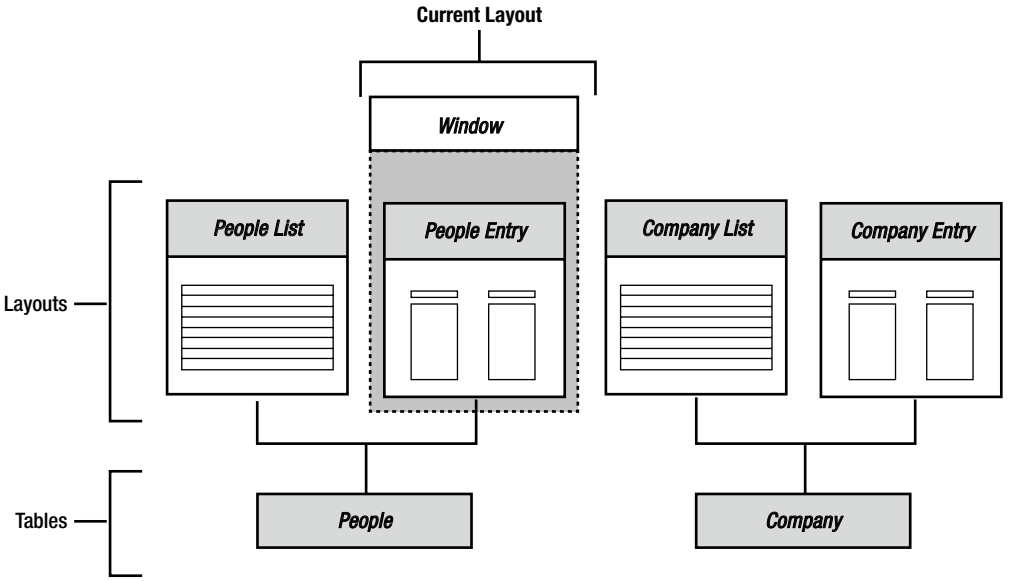

Figure 1-3. *Illustrating layouts displaying data from their corresponding tables with the window displaying the current layout*

A *found set* of records is a subset of the total records contained in the table being displayed in a window, typically after a search. The *current record* is whichever record from the found set is displayed within a window showing one record or whichever record is selected within a window showing a list of records.

Scripted Procedures

A *script* is a stored set of actions that can be automatically performed in response to user activity or on a schedule. Scripts can generally perform any function with which a user has access.

Identifying Different Database Architectures

FileMaker employs a uniquely *integrated architecture* that skillfully manages to maintain an intuitive environment that allows most users to acclimate regardless of their level of development experience. However, it may be somewhat foreign to experienced developers who are familiar with traditional multitier architecture of database systems like SQL. Ironically, this may cause those with *more* database development experience to have a greater learning curve with certain FileMaker concepts.

The following comparison provides an overview sketch of the situation and may help any developer new to FileMaker better understand the integrated nature of the application and the database file structure it is used to create.

The Traditional Multitier Database Architecture

Most databases are built using a multitier architecture. For database software engineers, a *multitier architecture*, illustrated in Figure 1-4, is an arrangement in which three functions — presentation front-end interface, process logic middleware, and back-end data — are usually physically separated and often built with different languages.

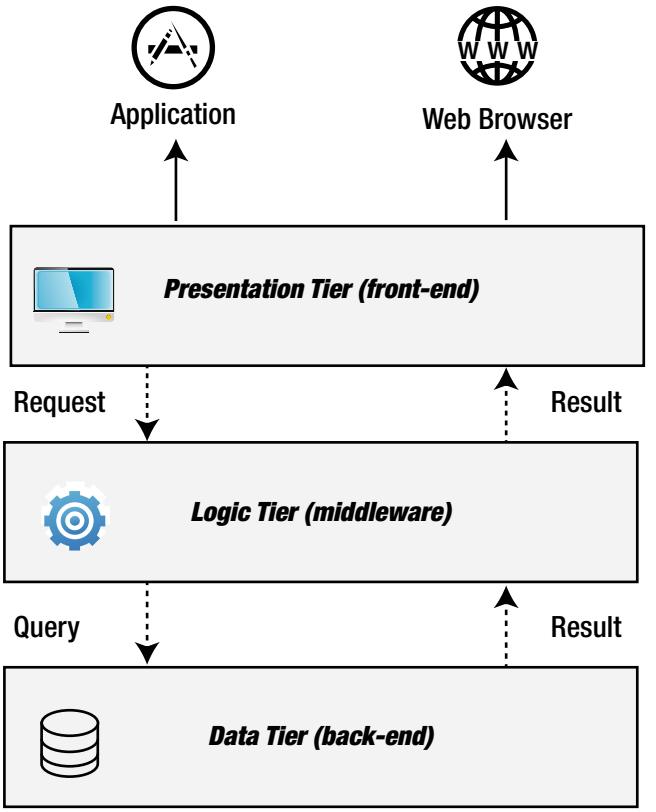

Figure 1-4. *Showing an overview of the traditional multitiered database architecture with data, interface, and script as separate but interacting systems*

Presentation Tier (Front End)

The *Presentation Tier* is a front-end interface. This top layer is where information is displayed in human readable form specifically for user interaction; data is viewed and edited on layouts and scripted procedures are accessible through menus and buttons.

Examples of a presentation tier include the following:

- A web page that includes database elements.
- A desktop or mobile application that presents data via windows and layouts.

Process Logic Tier (Middleware)

The *Process Logic Tier* is the middle layer that contains business logic: code that does much of the work to process commands, make decisions, evaluate requests, format results, and generally facilitate processes between the interface and the stored data.

Examples of the logic tier include these:

- PHP
- Java

Stored Data Tier (Back End)

The *Stored Data Tier* is a back-end data system. This bottom layer is generally hidden out of sight, which handles information storage and retrieval from a database or file system.

Examples of the stored data tier include these:

- Oracle databases
- MySQL database

FileMaker's Integrated File Architecture

Unlike the distributed, multitiered design, FileMaker is an *integrated file-based architecture, illustrated in Figure 1-5*. An *integrated architecture* is one in which all components that make up a database are built and stored together into a single file or system that is accessed all at once.

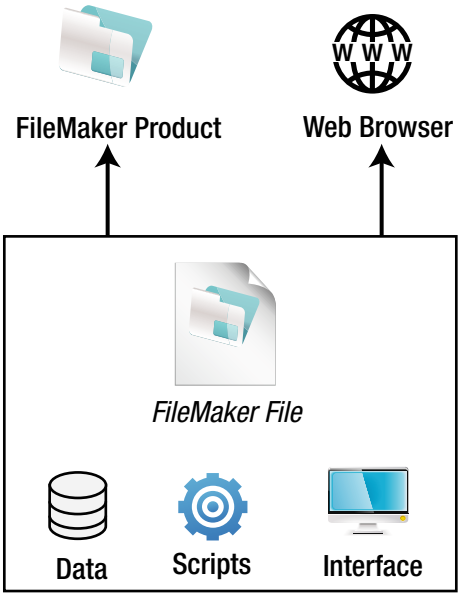

Figure 1-5. *An overview of FileMaker's integrated database architecture with data, interface, and script combined in a single file*

While FileMaker's integrated file structure is most likely inherited from its early development as a simple, end-user, file-building tool — this characteristic has been retained as it has evolved into the modern powerhouse it is today. Newer features that allow external data sources, including SQL tables, to be pulled into a FileMaker database and used as if they were internal, result in a flexibility that provides the best of both worlds.

Some benefits of this approach include the following:

- Develop the entire database system from within a single, easy-to-use application.
- Databases are files and can be hosted from a central server or a desktop, or used and shared like self-contained, portable packages just like any other document on a computer.
- Since a database can include and use external data sources, it is possible to approximate a multitiered structure. For example, some developers store data tables in a file separate from the user interface and scripted processes.

Taking a Bird's Eye Anatomical and Functional Overview

FileMaker's unique integration doesn't stop at the file structure. Each version of the desktop application — FileMaker Pro and FileMaker Pro Advanced — provides a slight variation on a single, complex but remarkably intuitive and uncluttered interface that ingeniously blends all functions for the development *and* use of feature-rich database files, including the following:

- Developing back-end data structures
- Developing front-end interfaces
- Developing scripted processes
- Providing front-end user access to the database

This seemingly impossible feat is accomplished by the skillful use of different window modes for specific functions, context-sensitive menus, and separate windows for certain development functions. This seamless and almost artfully designed environment obscures an enormous complexity, which can present a challenge for both new users and authors of books such as this one. Therefore, a bird's eye overview of the many components involved and topics to cover herein may be beneficial.

The application resources and defined file components that work together to organize, store, and manage information are shown in Figure 1-6. Although there are points of overlap, the components can be roughly divided into five categories, split between the application and the database file.

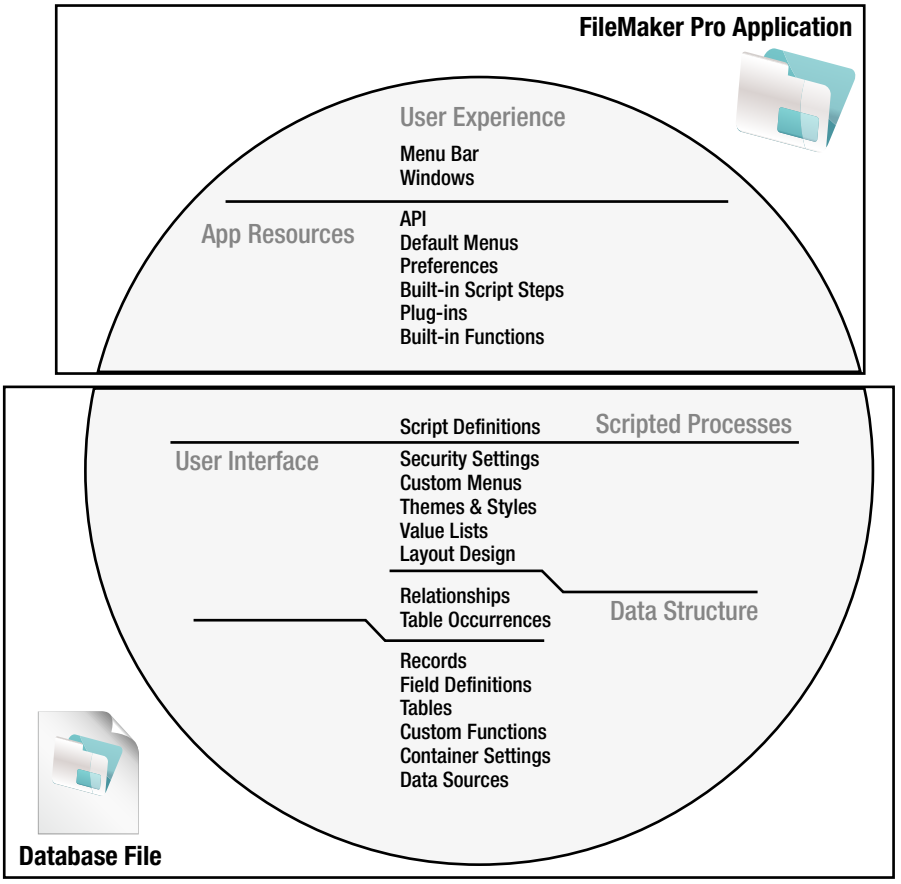

Figure 1-6. *The component breakdown between the FileMaker application and a database file, grouped by primary role*

The five major categories of components are these:
Application

- Application resources
- User experience

Database File

- Data structure
- User interface
- Scripted processes

Application Resources

The FileMaker application contains basic resources that interpret settings and information in a database file to generate a user experience.

Application Resources

Application resources are those things that make up the application programming interface (API) used to develop custom databases and create the development-user and end-user experience, including the following:

- Preferences (Chapter 2)
- Plug-ins (Chapter 38)
- Default menus (Chapter 2)
- Built-in functions (Part 3 — or — Chapters 13-24)
- Built-in script steps (Chapters 34 and 35)
- Sharing and networking (Chapters 39-41)

User Experience

The *user experience* covers what the user sees and how they interact with a database in general use. This includes the menus (default or custom) and the windows that the application presents to the user in the interface. The user can then take actions, including these:

- Viewing an interacting with database windows (Chapter 3)
- Creating, opening, and navigating a database file (Chapter 3)
- Working with records (Chapter 4)
- Importing and exporting data (Chapter 5)

Database File Resources

A *database file* contains everything unique to your project: all the settings for the data structure, user interface and scripted processes, as well as the records of data entered to date.

Data Structure

The *data structure* portion of a database contains all the settings that define what is stored in the file, how it connects, and what functions can be performed on it, and the data itself, including the following:

- Tables (Chapter 7)
- Fields (Chapter 8)
- Data sources, table occurrences, and relationships (Chapter 9)
- Calculation formulas (Chapter 12)
- Custom functions (Chapter 25)

- SQL queries (Chapter 26)
- Container settings (Chapter 10)
- Value lists (Chapter 11)

User Interface

The *user interface* of a database contains all the settings that define the way things are displayed on layouts in windows, including the following:

- Layouts (Chapters 27-31)
- Themes and styles (Chapter 32)
- Custom menus (Chapter 33)
- Security settings (Chapter 40)

Scripts

The *scripts* of a database define the procedures that can be performed automatically or at a user's request. (Chapters 34-37).

The FileMaker Product Family

The FileMaker Pro 16 product family is made up of five different applications, shown in Figure 1-7.

Figure 1-7. *The FileMaker 16 product family lineup*

Each of these applications has a unique feature set and is either purpose or device specific in its ability to develop, use, and host databases as listed in Table 1-2.

Table 1-2. *The supported devices and fundamental features for each product in the FileMaker family*

Product Name	Platform(s)	Develop	Use	Host
FileMaker Pro	Mac, Windows	Yes	Yes	Yes
FileMaker Pro Advanced	Mac, Windows	Yes	Yes	Yes
FileMaker Server	Mac, Windows	No	No	Yes
FileMaker Go	iOS Devices	No	Yes	No
FileMaker WebDirect	Web Browser	No	Yes	No

FileMaker Pro

The standard desktop application is named *FileMaker Pro*. This application provides general development features, including the following:

- *User Access* — provides intuitive front-end access to database features.
- *Import/Export* — easily exchange data between a FileMaker database and `CSV`, `Tab`, `XML`, `ODBC`, Microsoft Excel files, other FileMaker databases, and more.
- *Customize* — a graphical development environment makes it easy to create custom databases.
- *Reporting* — easily build layouts for summarizing data in reports or charts.
- *Automate* — create scripts to repeat a sequence of actions automatically.
- *Sharing* — easily share a database with other users by turning on peer-to-peer networking or uploading a database to a FileMaker Server.
- *Secure* — create accounts and passwords to limit user acess and activity.
- *Integrate* — connect external `SQL` data sources and interact with them; create custom workflows with scripting languages that allow interaction with other applications using AppleScript (`macOS`) and Dynamic Data Exchange (`Windows`).

FileMaker Pro Advanced

The developer version of the desktop application is called *FileMaker Pro Advanced*, which has all the features of the standard version plus a handful of additional features intended for advanced developers, including the following:

- *Custom Functions* — allow the creation of your own functions.
- *Custom Menus* — allow the FileMaker menu to be overridden and replaced with custom menus and items that perform custom scripts.
- *Data Viewer* — easily monitor fields, variables, and calculations while troubleshooting your custom database.
- *Database Design Report* — generate a detailed report of all the elements that make up a database.

- *Database Encryption* — enable AES 256-bit encryption to protect a shared database.
- *Multiple Table Import* — enhanced ability to import multiple tables from one database to another.
- *Script Debugger* — helps to identify script problems and evaluate performance problems.

■ **Tip** To follow along as we explore many of the advanced subjects discussed in this book, you will want a copy of FileMaker Pro Advanced installed on your computer.

FileMaker Server

FileMaker Server is database-hosting software that is based on industry-standard security technology to offer user-scalable, secure, and reliable round-the-clock access to databases for authorized users. Its features include the following:

- *Host Databases* — dedicated server allows simultaneous, multi-user access to databases from across a network by users of FileMaker Pro, FileMaker Go, and FileMaker WebDirect.
- *Media Streaming* — container field content uses progressive downloading technology where applicable to begin streaming content for immediate display. For example, a user will not have to wait for a movie stored in a container field to fully download before it begins playing.
- *Scheduled Events* — allow various events to be scheduled for automatic execution including database backups, run system-level scripts (shell script or batch file), and run FileMaker script in a FileMaker database.
- *Perform Script on Server* — databases can opt to have specific scripts run from the server rather than on a user's computer to offload tasks and save time.
- *Logging* — administrators can access event and error logs to troubleshoot problems and to help optimize database performance.
- *Web Direct* — authorized users can access databases securely across the Web through a browser window.

■ **Note** `FileMaker Cloud` is another option for hosting databases on the Internet for wide area sharing. The service combines Amazon Web Services (AWS) and FileMaker Server software (Chapter 39).

FileMaker Go

FileMaker Go is a free `iOS` app available for both the iPad and iPhone that allows users to access database resources when away from their desktop computer. With the app, users can access databases stored locally on their `iOS` devices or from a FileMaker Server across a network. The app does not include an API, so databases must be developed on the desktop application prior to use on a device.

■ **Note** Accessing a database hosted on a `FileMaker Server` from `FileMaker Go` or with `FileMaker WebDirect` requires an available user connection license for each connection (Chapter 39: Understanding Client License Options and Limitations).

FileMaker WebDirect

Although FileMaker offers other options for web hosting, deploying a database on the Web is a lot easier with *FileMaker WebDirect*. Once a database has been built with the desktop application, it can be easily enabled for web sharing and hosted on a FileMaker Server. Users can then access the database through a web browser. Layouts are automatically rendered in the browser and, except for certain incompatible script steps and the ability to work in multiple windows, most functionality is identical to that on a desktop or iOS device.

Summary

In this chapter, we discussed an overview of FileMaker and basic database concepts. We looked at the big picture history, architecture differences, anatomical overview of components, and we reviewed the FileMaker product line.